inalco
Institut national
des langues
et civilisations orientales

SoftBank
Robotics

Institut National des Langues et Civilisations Orientales

Département Textes, Informatique, Multilinguisme

Response Generation in a Dialogue System: Bouncing Back with Word Embeddings

par

Noor Alkhadhar

Master

Traitement Automatique des Langues

Recherche et Développement

Directeur de mémoire :
Mathieu Valette

Encadrant de stage :
David Houssin

Année universitaire 2017–2018

Je remercie
Mathieu Valette
David Houssin, et
Yufo Fukuda,
pour leur
présence,
soutien, et
souplesse.

Contents

List of Figures

List of Tables

Introduction

This experimental research project seeks to gain a better understanding of word embeddings generated in a neural network layer. More precisely, we will attempt to use these word vectors to construct a concept association mechanism in a dialogue system, allowing it to bounce off previous input in order to spontaneously generate new responses. The motivation behind this work is the desire to render a dialogue system robust enough to be able to handle input sequences that are not accounted for in its database of rules.

Part I

Background

CHAPTER 1

WORD EMBEDDINGS

Contents

1.1 Introduction

In this first chapter, we will be discussing word embeddings in detail: from their origins in linguistics, to their use in Natural Language Processing (NLP) via the Deep Learning community. We will also go through the main types of word embedding models: count-based models and predictive models. The latter category subdivides further into explicit and implicit models, which differ in the way they calculate their vectors. We will then survey some of the most common machine learning algorithms used today for generating word embeddings: `word2vec`, gloVe, and fastText.

Before delving into the subject matter, a clarification is due. Throughout the years and the scientific disciplines, several terms have appeared to refer to what is virtually the same thing: a *Distributional Semantic Model* (DSM). These distributed representations of words, essentially co-occurrence matrices of words built from large quantities of text, have been used as components in different models, be it in continuous space language models or as a way to study the relations between words based on their co-occurrences in a given corpus. Hence, DSMs have been variously called semantic spaces, continuous vector representations of words, continuous space word representations, context-predicting models, and even neural language models,

depending on the method used to compute them. We will principally be using the terms *word embeddings* and *vector space* throughout this study.

Put simply, in a word embedding space, each word is converted into a vector, and the aim is to represent the word's meaning mathematically through its vector. In this sense, similar words should have similar vectors, and relations between vectors should encode some kind of linguistic information between the words they represent. This has been attempted in several different ways, which we will now explore.

1.2 History

1.2.1 Count-based Models

Within the field of linguistics, word embeddings were first developed in the subfield of distributional semantics, a research area whose core concept is defined by the **Distributional Hypothesis** (Harris, 1954):

"linguistic terms with similar distributions have similar meanings"

The main objective of distributional semantics is therefore defined as the classification of semantic similarities between words based on distributional properties determined by examining large quantities of text data.

Traditionally, DSMs were built by *counting* the contexts in which a certain word appears in a corpus and stocking them in a vector, that vector becoming a sort of meaning representation for the word. Since these DSMs built purely on co-occurence counts had consistently shown low performance on several tasks (Baroni, Dinu, & Kruszewski, 2014), two main methods have been applied in the aim to improve their performance: using different weighting schemes for additional context information, and dimensionality reduction techniques to compress the vectors.

With regard to using different weighting schemes in an attempt to improve the performance of a count-based DSM, we add contextual informativeness by giving more weight to less frequent words in the vocabulary. To illustrate, it is more meaningful if a word co-occurs frequently with a rare noun or verb, rather than with a stop word such as *the* or *is*. Two weighting schemes commonly used by the scientific community are *positive Pointwise Mutual Information* (positive PMI) and *Local Mutual Information*. (Evert, 2005)

Word vectors can also be compressed in order to reduce the number of dimensions they contain. This is commonly done through *Singular Value Decomposition* (SVD) (Golub & Van Loan, 1996) and *Non-negative Matrix Factorization* (Lee & Seung, 2000). This vector optimisation process, composed of adjusting weights within the vectors and reducing vector dimensionality, is considered an unsupervised process. However, it is usually indirectly supervised because several different settings

are tested and evaluated, and the best parameter setting is then selected depending on the task at hand.

1.2.2 Predictive Models

With the appearance of the *Vector Space Model* (VSM) in the 1960s (Dubin, 2004) first came the idea of using vectors to represent words mathematically. Specifically in NLP, words were first represented as one-hot vectors where the number of columns in the vector is equal to the size of the vocabulary, and the values for all of these columns are 0, except for a single 1 in the column of the word that a given vector is meant to represent. In this way, words are treated atomic units, and vectors encode no information whatsoever on word similarity and how words interact.

This method was successful in representing documents as vectors for tasks such as document filtering, classification, retrieval, and selection for query answering. One of the most widely used schemes for calculating term weights in a document vector is *tf-idf*, which stands for *term frequency-inverse document frequency*. Simply put, given a corpus of documents, the tf-idf value increases with the frequency of a certain term in a document, and decreases as the number of documents in the corpus that also contain the term rises. (Jones, 1972 ; Luhn, 1957) In this way, we can take into account that more frequent terms, such as stop words, generally tend to appear more often in documents, and adjust for this by giving heavier weights to rarer terms.

This simple method of representing words as one-hot vectors, trained on large quantities of data, yielded better results than more complex systems trained on smaller quantities of data. For instance, in statistical language modelling, the n-gram model (Brants, Popat, Xu, Och, & Dean, 2007) employed this method of representation of words.

Successful in some domains, there are limits to this simple technique of words represented as one-hot vectors. Although it is possible to train n-grams on large quantities of text data (trillions of words as shown by Brants et al. (2007)), there are other domains, such as machine translation (MT) and automatic speech recognition (ASR), for which the available training data do not contain enough words for these simple word representation techniques to be effective.

With the rise of machine learning and particularly deep learning in NLP came the possibility to train more complex models on a much larger quantity of text data. These much more complex neural network language models (NNLMs), were shown to vastly improve on results obtained using n-gram models (Bengio, Ducharme, Vincent, & Janvin, 2003 ; Mikolov, Karafiát, Burget, Cernocký, & Khudanpur, 2010 ; Schwenk, 2007). The results achieved by these models were indeed impressive and improved on previous methods, but the depth of the models meant that the way the

results were achieved remained a mystery, a black box. Their accuracy led some to believe that the model actually understood natural language. Therefore, when NNLMs first appeared, they caused a stir in the field of NLP with the belief that they were a solution to many long-standing NLP problems.

Within an NNLM, high-dimensional real valued vectors, which have been converted from words through a learned lookup table, are generated in one layer—the *embedding layer*—and used as inputs to the neural network. In this way, the embeddings are a feature of the model. These word embeddings, learned from unlabelled text data, marked a significant change in the means of production of representations of words in a vector space. Furthermore, these distributed representations of words (G.E. Hinton, 1986) **proved to be more successful than one-hot vectors that stored words simply as indices in a vocabulary**.

It has been shown that the word embeddings generated in a layer of the NNLM can have applications in several unrelated NLP tasks (Collobert & Weston, 2008 ; Turian, Ratinov, & Bengio, 2010). That being said, an NNLM is very costly to train, taking weeks to months depending on the amount of training data (Mikolov, Chen, et al., 2013). If we remove all other, computationally expensive, components of the NNLM besides the layer in which the word embeddings are produced, we could theoretically create a highly optimised shallow machine learning algorithm for producing word embeddings from text data.

Thus came into existence this new generation of DSMs, which set the vector weights directly during model training as the vectors are being constructed. So, unlike count-based models, which first collect context vectors for the words and then perform vector transformation to re-weight the vectors, these new predictive models combine this into one step during which the vector weights are set to maximise the probability of a word appearing in a certain context given a co-occurence window of n history and future words *or* to best predict the context in which the word has a tendency to appear in throughout the corpus—depending on the neural network architecture being used.

These context-predicting vectors were initially simply a by-product of deep neural network language models; they were a result of the embedding layer and were then fed into the model as input. Seeing as they were developed within the machine learning community, a lack of interdisciplinary awareness or communication meant that researchers in the field had little or no knowledge of other DSM work in computational linguistics, namely the context-counting vectors we have discussed. Because of this, there had been no previous work thoroughly comparing the performance of these two types of models until Baroni et al. (2014) who concluded that predictive

models consistently yield results superior to count models.

Several different shallow machine learning algorithms have been developed for producing word embeddings, given a corpus of text data. In order to better understand the behaviour of predictive models, the next section will examine some of the most popular word embedding algorithms of this type.

1.3 Methods

1.3.1 `word2vec`

Mikolov, Chen, et al. (2013) released, with Google, the `word2vec` toolkit for learning word embeddings. One of the reasons they developed this toolkit was to show that shallow neural networks can be used to generate very good word embeddings. It was even argued that a shallow architecture such as that of `word2vec` is preferable to deep networks because they are easier to optimise. This made it possible to train the optimised shallow models on even more data much more quickly than NNLMs. Mikolov also wished to debunk the belief that the NNLM understands human language, by showing that the word embedding space encoded linguistic information within the compositionality of its word vectors. We will explore this concept in a more in-depth manner in Chapter 2.

By developing the `word2vec` toolkit, Mikolov, Chen, et al. (2013) highlighted the danger of viewing deep learning as a solution for all NLP tasks, because the depth of these models may cause the misleading view that the machine understands and has learned to manipulate natural language. The authors also cut down the time required to train the model and generate the word embeddings tremendously: from several weeks to several days. In addition to this remarkable optimisation of training time, `word2vec` is a much more computationally economic solution, dissolving the necessity of training on a supercomputer to generate word embeddings.

The architecture of `word2vec` is an extension of Mikolov (2007) ; Mikolov, Kopecky, Burget, Glembek, et Cernocky (2009). This work followed the method mentioned previously, which essentially consists of breaking down a complex deep neural network language model into two steps: first learning the word vectors using a neural network with one hidden layer, and then training an NNLM using these word vectors as input. The `word2vec` algorithm develops this first step of learning the word vectors using a shallow neural network.

Having noticed that the complexity in both feedforward and recurrent NNLMs stems mainly from their non-linear hidden layer, the authors of this paper present two new log-linear models for computing word vectors. With these new model architectures, they aimed to maximise accuracy while minimising computational complex-

ity as compared to NNLMs.

Continuous Bag-of-Words Model

In the feedforward NNLM, the most computationally expensive layer is the hidden one in which the output word's probability distribution over all the words in the vocabulary is computed by a softmax classifier. The `word2vec` *continuous bag-of-words* (CBOW) model optimises this layer significantly by using hierarchical softmax instead. In this way, the non-linear hidden layer of the feedforward NNLM is removed, and all words of the vocabulary are projected into the same position in the projection layer—instead of only the projection matrix. (For a detailed explanation of

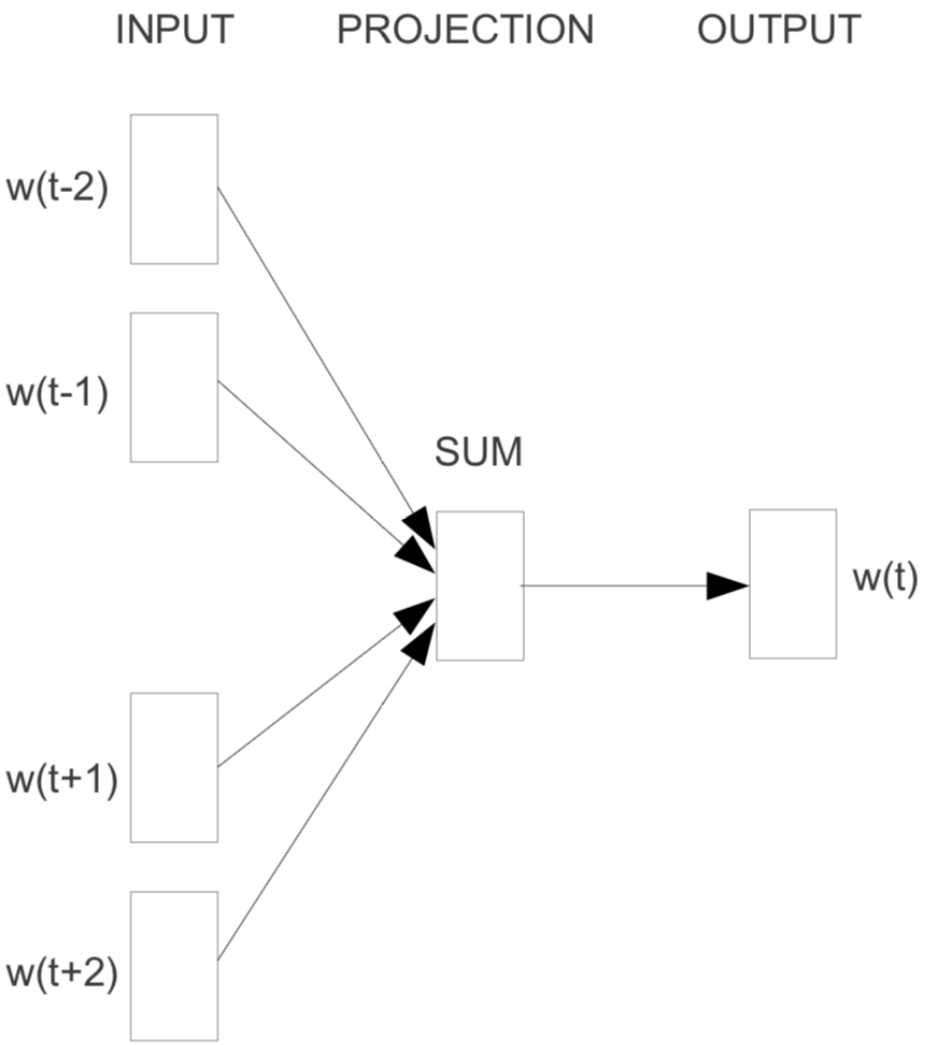


Figure 1.1 – CBOW model architecture (Mikolov, Chen, et al., 2013)

the structure of a feedforward NNLM, see section 2.1 in Mikolov, Chen, et al. (2013))

This model is called *bag-of-words* because the order of words does not affect the projection, seeing as the projection layer is shared by all words, and *continuous* because it uses continuous distributed representation of the context. The CBOW model is a two-layer neural network, as illustrated in Figure 1.1: the input layer, a projection layer, and an output layer. Given a context window of n words, the model tries to predict the word that would appear in that context. The word embeddings are calculated in the projection layer.

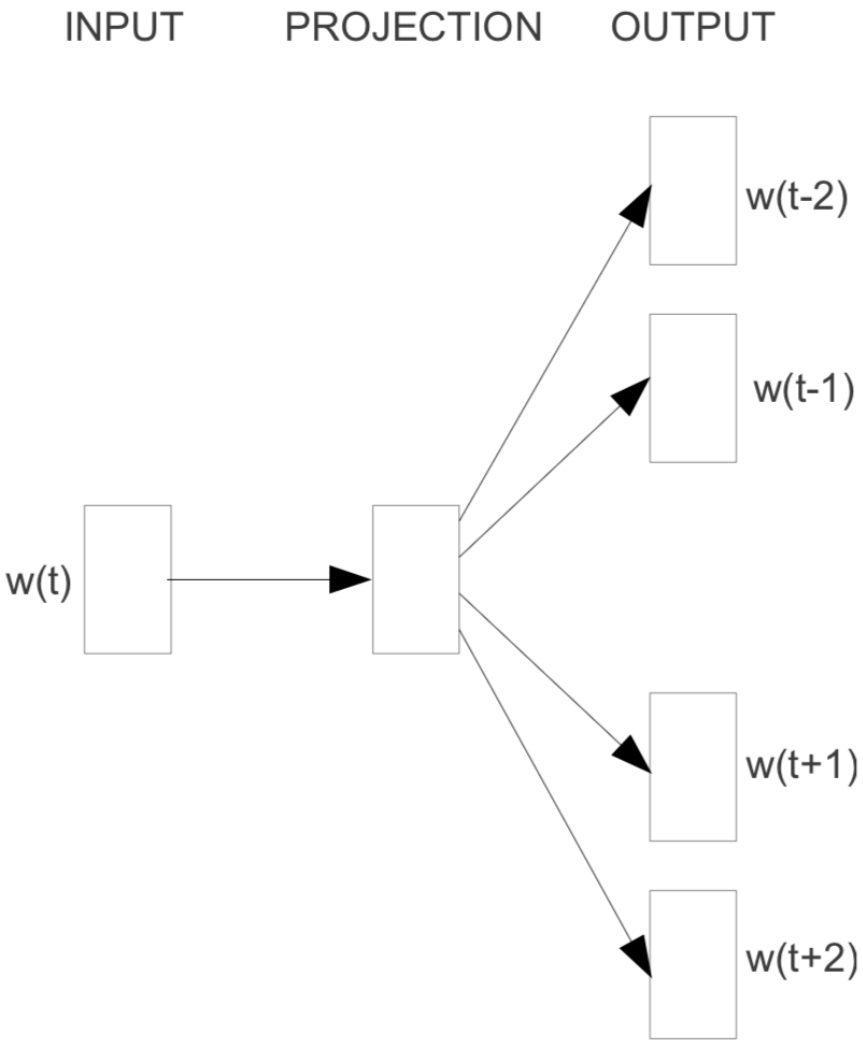


Figure 1.2 – Skip-gram model architecture (Mikolov, Chen, et al., 2013)

Continuous Skip-gram Model

The second model presented in the paper is the *skip-gram* model which, inversely to the CBOW model, takes in a single word as input and tries to predict the context in which it would appear. The context range is defined by the model parameters; increasing the range increases accuracy of the vectors generated, at the expense of higher computational complexity. It is worth noting that not all words in the given range are considered; rather, given a range n, the model will select a number r at random between 1 and n and consider as correct labels r words before and r words after the input word.

Figure 1.2 shows the architecture of the skip-gram model, composed of two layers like the CBOW model, but differing from CBOW in that the input layer takes in a single word and tries to correctly predict surrounding words.

1.3.2 GloVe

The development of GloVe (Global Vectors for Word Representation) by Pennington, Socher, et Manning (2014) divides the category of predictive models into two subcategories: implicit models such as `word2vec`, and explicit models, such as GloVe. This type of model is called explicit because it claims to make explicit the features needed in order to correctly encode linguistic information in the vector representations of words. It combines global matrix factorisation methods from models like *Latent Semantic Analysis* (LSA) and local context window methods from models such as the `word2vec` skip-gram model.

The authors of the article argue that both of these models are lacking: LSA performance on the word analogy task (explained in Chapter 2) suggests "sub-optimal vector space structure", and the skip-gram model fails to leverage global statistics of the corpus—which may contain useful information—as it trains only on local context windows. By combining the two models, the authors claim that they can improve on the skip-gram architecture by using matrix factorisation methods and therefore taking into account the co-occurence statistics of the corpus as a whole, information that is not not taken advantage of by Mikolov, Chen, et al. (2013).

They conclude that, after a series of tests, GloVe performs better than `word2vec`. However, Levy et al. (2015) conduct a careful analysis of the performance of several embedding algorithms, and their results—in Figure 1.3—show higher accuracy, shorter training time and less memory consumption for `word2vec` than for GloVe.

1.3.3 fastText

The fastText library, released by Facebook Research (Bojanowski, Grave, Joulin, & Mikolov, 2016), builds on `word2vec` by including character n-gram information

win	Method	WordSim Similarity	WordSim Relatedness	Bruni et al. MEN	Radinsky et al. M. Turk	Luong et al. Rare Words	Hill et al. SimLex	Google Add / Mul	MSR Add / Mul
2	PPMI	.732	**.699**	.744	.654	.457	.382	.552 / .677	.306 / .535
	SVD	.772	.671	**.777**	.647	**.508**	.425	.554 / .591	.408 / .468
	SGNS	**.789**	.675	.773	**.661**	.449	**.433**	.676 / **.689**	.617 / **.644**
	GloVe	.720	.605	.728	.606	.389	.388	.649 / .666	.540 / .591
5	PPMI	.732	**.706**	.738	**.668**	.442	.360	.518 / .649	.277 / .467
	SVD	.764	.679	**.776**	.639	**.499**	**.416**	.532 / .569	.369 / .424
	SGNS	**.772**	.690	.772	.663	.454	.403	.692 / **.714**	.605 / **.645**
	GloVe	.745	.617	.746	.631	.416	.389	.700 / .712	.541 / .599
10	PPMI	.735	**.701**	.741	.663	.235	.336	.532 / .605	.249 / .353
	SVD	.766	.681	.770	.628	**.312**	.419	.526 / .562	.356 / .406
	SGNS	**.794**	.700	**.775**	**.678**	.281	**.422**	.694 / .710	.520 / **.557**
	GloVe	.746	.643	.754	.616	.266	.375	.702 / **.712**	.463 / .519
10	SGNS-LS	.766	.681	**.781**	**.689**	**.451**	.414	.739 / **.758**	.690 / **.729**
	GloVe-LS	.678	.624	.752	.639	.361	.371	.732 / .750	.628 / .685

Figure 1.3 – Performance of each method across different tasks (Levy et al., 2015)

of the word as additional input in the first layer of the neural network—an innovative characteristic that is lacking in other models such as `word2vec` and gloVe. This means that fastText considers all character subsequences—within a set length range—when computing the vector representation of a given word.

Where in the classic models, *dark* theoretically does not have anything in common with *darker* or *darkest*, subword information in the fastText model allows us to account for this relation. The subword information also enables the model to effectively handle out-of-vocabulary words by constructing their vector using the vectors of subsequences. Additionally, we can compute better vectors for rare words that do not appear in many contexts in the corpus by using the contexts of related words.

Model	Analogy	RW	Squad
GloVe Wiki + news	72	0.38	77.7%
fastText Wiki + news	**87**	0.50	78.8%
GloVe Crawl	75	0.52	78.9%
fastText Crawl	85	**0.58**	**79.8%**

Figure 1.4 – Comparing GloVe and fastText results on Word Analogy, Rare Words, and Squad datasets (Mikolov et al., 2018)

When it comes to enriching the word embeddings with character n-grams, the real gain is seen in morphologically rich languages, such as Czech and Russian, for which performance of the model increased with the addition of subword information. There is also a slight gain for languages such as Spanish and French. A study conducted

by Mikolov et al. (2018) shows that fastText outperforms GloVe word vectors on the Word Analogy, Rare Word and Squad data sets when trained on comparable text data sets (Figure 1.4).

1.4 Summary

In this section, we have seen that word embeddings were first built in computational linguistics and called distributional semantic models. Count-based models were created by counting the co-occurrences of a given word in a corpus, and then optimising the vectors through re-weighting schemes and vector reduction methods. Predictive models use a shallow neural network architecture to smooth out these two steps into one, the weights being calculated directly in the projection layer so as to maximise the probability of a word given a certain context (CBOW) or the probability of surrounding words given an input word (skip-gram).

Having learned about the history of word embeddings as well as the main types of models and algorithms used to produce them, **how are the word embedding spaces generated by these different algorithms evaluated?** One popular metric is the word analogy task.

CHAPTER 2

ANALOGY RESOLUTION

Contents

2.1 Introduction

Now that we are familiar with word embeddings and how they are generated using different methods, we will examine the ways in which we can evaluate them. The initial goal of word embedding evaluation was to ensure that they were calculated correctly during training of the model. In Mikolov's terms: we want the word embeddings to not only represent similarities between words, but also *different types of similarities* between words. For instance, the word embeddings should somehow encode the relationship between *Paris* and *France*—Paris being the capital of France—but also the relationship between *Paris* and *Prague*, as they are both European capitals.

This is shown in Figure 2.1, where in the left panel we see vector offsets capturing the male-female relation, and in the right panel we see how multiple relations can

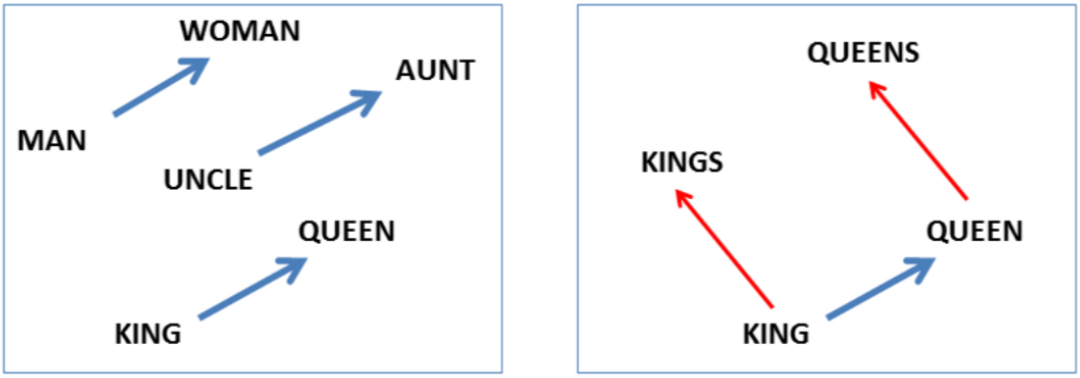


Figure 2.1 – Multiple relations can be embedded for a single word in a high-dimensional vector space (Mikolov, Yih, & Zweig, 2013)

be embedded in the vector for a single word. In this case, the projection shows the male-female relation as well as the singular-plural relation.

To illustrate this, the traditional example in the field is to show that gender information is encoded in the vector space, as performing vector arithmetic on the vectors of *king*, *man* and *woman* results in a vector very close to *queen*. If the vector of a word i is denoted by x_i, this relationship can be formulated as follows:

$$x_{king} - x_{man} + x_{woman} \approx x_{queen}$$

In natural language, the analogy $x_{man} : x_{king} :: x_{woman} : x_{queen}$ can be expressed as "***man*** *is to* ***king*** *as* ***woman*** *is to* ***queen***".

In comes the word analogy resolution task, the most commonly used metric to evaluate word embeddings. Several test sets have been developed to this end, and experiments conducted to compare word embeddings produced by different models, or by the same model but with varying hyperparameter settings. In this section, we will first present a few analogy test sets currently available. Next, we briefly outline the word analogy task for word embeddings, and proceed to examine the most successful methods for analogy resolution and the results of recent studies comparing these methods. Finally, we will discuss potential issues with the word analogy task and its widespread use as a generic evaluation metric for word embeddings.

2.2 Analogy test sets

2.2.1 SemEval-2012 (Jurgens et al., 2012)

The SemEval-2012 analogy test set is pulled from data from the relational similarity task in SemEval-2012, which included relations between word pairs targeting a large number of semantic relationships. It contains 79 fine-grained word relations,

Subcategory	Relation name	Relation schema	Paradigms	Responses
8(e)	AGENT:GOAL	"Y is the goal of X"	pilgrim:shrine assassin:death climber:peak	patient:health runner:finish astronaut:space
5(e)	OBJECT:TYPICAL ACTION	"an X will typically Y"	glass:break soldier:fight juggernaut:crush	ice:melt lion:roar knife:stab
4(h)	DEFECTIVE	"an X is is a defect in Y"	fallacy:logic astigmatism:sight limp:walk	pimple:skin ignorance:learning tumor:body

Figure 2.2 – Example test set patterns of the SemEval-2012 Task 2 (Jurgens et al., 2012)

such as *class–inclusion*, *part–whole* and *cause–purpose*. Figure 2.2 shows some examples of relations and responses generated by participants in the task.

2.2.2 Microsoft Research (Mikolov, Yih, & Zweig, 2013)

Created by Mikolov, Yih, et Zweig (2013), this is one of the first analogy test sets. It contains analogy questions in the form "a is to b as c is to ___". It is a purely syntactic test set, and the syntactic regularities that it aims to test are shown in Figure 2.3. For each pattern, the word pairs appear in both orders in the test set, meaning that if the test set contains "*see:saw::return:*___", it will also contain "*saw:see::returned:*___".

Category	Relation	Patterns Tested	# Questions	Example
Adjectives	Base/Comparative	JJ/JJR, JJR/JJ	1000	good:better rough:___
Adjectives	Base/Superlative	JJ/JJS, JJS/JJ	1000	good:best rough:___
Adjectives	Comparative/ Superlative	JJS/JJR, JJR/JJS	1000	better:best rougher:___
Nouns	Singular/Plural	NN/NNS, NNS/NN	1000	year:years law:___
Nouns	Non-possessive/ Possessive	NN/NN_POS, NN_POS/NN	1000	city:city's bank:___
Verbs	Base/Past	VB/VBD, VBD/VB	1000	see:saw return:___
Verbs	Base/3rd Person Singular Present	VB/VBZ, VBZ/VB	1000	see:sees return:___
Verbs	Past/3rd Person Singular Present	VBD/VBZ, VBZ/VBD	1000	saw:sees returned:___

Figure 2.3 – Test set patterns of the Microsoft Research analogy test set (Mikolov, Yih, & Zweig, 2013)

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Figure 2.4 – Categories and examples of the Google analogy test set (Mikolov, Chen, et al., 2013)

2.2.3 Google (Mikolov, Chen, et al., 2013)

This test set is comprised of 9 categories of syntactic questions and 5 categories of semantic questions, shown in Figure 2.4 alongside two examples from each of these categories. Each category is made up of 20–70 unique word pairs. This adds up to 10,675 syntactic questions and 8,869 semantic questions. The Google analogy test set contains only single token words, meaning multi-word entities such as *San Francisco* are not included.

2.2.4 BATS (Gladkova et al., 2016)

This test set was developed by Gladkova et al. (2016) in an aim to create a more balanced analogy test set. The *Bigger Analogy Test Set* (BATS) is made up of 99,200 questions in total, across 40 semantic and morphological categories. The creators of BATS argue that existing test sets focus only on one category or are unbalanced. Indeed, the SemEval-2012 analogy test set contains only semantic questions, the Microsoft Research analogy test set contains only morphological questions, and the Google analogy test set contains 9 morphological and 5 semantic categories, with anything from 20 to 70 word pairs per category.

Due to the lack of uniformity across different analogy test sets, the creators point out the importance of reporting results on all categories and relations, and not only

the average accuracy of a model. This, however, is not common practice in the field, hence the importance of a balanced analogy test set such as BATS, which includes 40 linguistic relations as shown in Figure 2.5.

2.3 Methods for solving analogies

In natural language processing, the term *analogy relationship* between words is used to refer to any type of similarity shared by two pairs of words. This means that we are not exclusively referring to semantic traits shared by pairs of words, but also morphosyntactic ones. For example, "*Berlin is to Germany as Paris is to France*" holds as a proportional analogy just as much as "*table is to tables as flower is to flowers*" does. An analogy relationship between two word pairs can thus be formulated in the following way: *a:a*::b:b**, and the goal of the analogy resolution task is to find *b**.

Two main approaches exist for analogy resolution methods. One approach is the **pair-based** method that, given *a:a*::b:?*, aims to find *b**. In other terms, given a pair of words *a* and *a** representing a certain relation, and the *b* word of the second pair, the goal is to correctly determine *b**, in that *b:b** share the same relation as *a:a**.

The second school of methods follows the **set-based** approach to analogy resolution. These methods capitalise on the generalisation of a target relation, meaning that instead of representing the relation with a single pair of words, the relation is represented with a set of pairs, known as the *training set*. So, given a set of pairs that share the relation shown by *a:a**, the goal is to find *b** in *b:b** where *b* and *b** share the same relation as the pairs in the training set. It is important to note that the *b:b** pairs in the test set, the ones we are trying to solve for, are obviously *not* included in the training set for the relation.

2.3.1 Pair-based methods

3COSADD

Interestingly, the fact that word embeddings generated by a neural network even encoded these linguistic similarities in the first place came somewhat as a surprise to the researchers. Mikolov, Yih, et Zweig (2013) found that neural word embeddings encoded meaningful syntactic and semantic regularities between words of a corpus, and that this vector relation could be characterised by a *relation-specific vector offset*. For *a:a**, the vector offset is their difference: $x_{a^*} - x_a$

For instance, the relation between a word's singular and plural form can be defined by constant vector offsets between pairs of words. This means that $x_{socks} -$

	Subcategory	Analogy structure and examples
Inflections	Nouns	I01: regular plurals *(student:students)*
		I02: plurals - orthographic changes *(wife:wives)*
	Adjectives	I03: comparative degree *(strong:stronger)*
		I04: superlative degree *(strong:strongest)*
	Verbs	I05: infinitive: 3Ps.Sg *(follow:follows)*
		I06: infinitive: participle *(follow:following)*
		I07: infinitive: past *(follow:followed)*
		I08: participle: 3Ps.Sg *(following:follows)*
		I09: participle: past *(following:followed)*
		I10: 3Ps.Sg : past *(follows:followed)*
Derivation	No stem change	D01: noun+less *(life:lifeless)*
		D02: un+adj. *(able:unable)*
		D03: adj.+ly *(usual:usually)*
		D04: over+adj./Ved *(used:overused)*
		D05: adj.+ness *(same:sameness)*
		D06: re+verb *(create:recreate)*
		D07: verb+able *(allow:allowable)*
	Stem change	D08: verb+er *(provide:provider)*
		D09: verb+ation *(continue:continuation)*
		D10: verb+ment *(argue:argument)*
Lexicography	Hypernyms	L01: animals *(cat:feline)*
		L02: miscellaneous *(plum:fruit, shirt:clothes)*
	Hyponyms	L03: miscellaneous *(bag:pouch, color:white)*
	Meronyms	L04: substance *(sea:water)*
		L05: member *(player:team)*
		L06: part-whole *(car:engine)*
	Synonyms	L07: intensity *(cry:scream)*
		L08: exact *(sofa:couch)*
	Antonyms	L09: gradable *(clean:dirty)*
		L10: binary *(up:down)*
Encyclopedia	Geography	E01: capitals *(Athens:Greece)*
		E02: country:language *(Bolivia:Spanish)*
		E03: UK city:county *York:Yorkshire*
	People	E04: nationalities *(Lincoln:American)*
		E05: occupation *(Lincoln:president)*
	Animals	E06: the young *(cat:kitten)*
		E07: sounds *(dog:bark)*
		E08: shelter *(fox:den)*
	Other	E09: thing:color *(blood:red)*
		E10: male:female *(actor:actress)*

Figure 2.5 – Categories and examples of the Bigger Analogy Test Set (Gladkova et al., 2016)

$x_{sock} \approx x_{girls} - x_{girl}$, $x_{apples} - x_{apple} \approx x_{girls} - x_{girl}$, and so on. This linear vector offset method to analogy resolution is known as **3COSADD**.

Before applying the 3COSADD method, vector normalisation is performed—a common preprocessing step in machine learning. In this case, the vectors are normalised to *unit norm* which means that the sum of the square of each element of the normalised vector would equal *1*. This type of normalisation produces a vector that is also—besides *unit norm*—often referred to as a *unit vector* or *vector of length 1*.

Given the analogy relationship *a:a*::b:b**, we first find the corresponding embedding vectors x_a, x_{a^*}, x_b and then solve for *b** by computing $y = x_{a^*} - x_a + x_b$. Since it is possible that no word exists at exactly y, we use *cosine similarity* to find the embedding vector that has the largest similarity to y and output that vector. This can be summed up by the following equation:

$$x^* = \underset{x' \notin \{a, a^*, b\}}{\operatorname{argmax}} \cos(x', a^* - a + b) \tag{2.1}$$

where

$$\cos(v, w) = \frac{v \cdot w}{\|v\|\|w\|} \tag{2.2}$$

3COSMUL

The **3COSMUL** method was proposed by Levy et Goldberg (2014) in which they show that the 3COSADD method is equivalent to adding and subtracting cosine similarities. They propose replacing the addition and subtraction operations with multiplication and division of similarities.

$$x^* = \underset{x' \notin \{a, a^*, b\}}{\operatorname{argmax}} \frac{\cos(x', a^*)\cos(x', b)}{\cos(x', a)} \tag{2.3}$$

ONLY-B

Linzen (2016) discusses various methods for analogy resolution and tests them all on the same vector space in order to correctly compare their performance. One noteworthy method tested in the paper is **ONLY-B**, which reaches an accuracy score of 0.70 in the singular-plural category. Illustrated in Figure 2.6, this method simply returns the nearest neighbour of *b*, ignoring both *a* and *a**.

$$x^* = \underset{x' \notin \{a, a^*, b\}}{\operatorname{argmax}} \cos(x', b) \tag{2.4}$$

2.3.2 Set-based methods

Drozd et al. (2016) showed that the issue with using 3COSADD to solve word analogy problems is its sensibility to word idiosyncrasy, and that we could improve accuracy by averaging over multiple word pairs to capture a certain relation. The

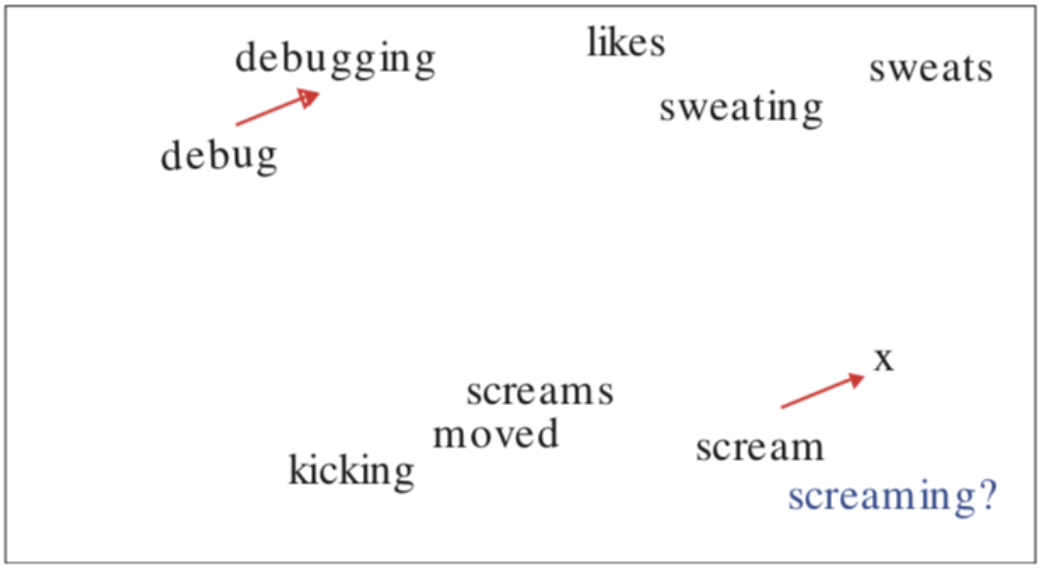


Figure 2.6 – When offsets are inconsistent or small, *b** may still be correctly returned if it is close to *b* (Linzen, 2016)

study also improves accuracy by 30% over the state-of-the-art by multiplying cosine similarity to a source word vector with logistic regression to determine to what extent the candidate answer belongs to the correct word class. These two new methods have been called **3COSAVG** and **LRCOS**, respectively.

3COSAVG

As previously stated, being based on a sole pair of words makes 3COSADD overly sensitive to noise and differences between words, which can be highlighted through polysemy networks. In the *king:man::queen:woman* example, *Queen* is also a musical group, which means it appears in many contexts *king* does not appear in. Furthermore, the gender characteristic may not be the only distinctive feature between them, depending on the corpus of text data used to generate the word embeddings.

To solve this problem, Drozd et al. (2016) propose **3COSAVG**, which learns the target relation from multiple word pairs instead of a single word pair. In this method, a list of word pairs representing the relation we wish to capture is divided into a training set and a test set, and we use the pairs in the training set to learn the target relation. This is what the authors call a "naive approach" because it is simply the average of the offsets between all pairs of the training set:

$$\underset{b^* \in V}{\operatorname{argmax}}(sim(b^*, b + AVG)) \tag{2.5}$$

where AVG represents the average offset of all pairs in the training set:

$$AVG = \frac{\sum_{i=0}^{m} a_i}{m} - \frac{\sum_{i=0}^{n} b_i}{n} \tag{2.6}$$

and a_i and b_i represent words from source and target classes.

LRCOS

The **LRCOS** method (Drozd et al., 2016) combines cosine similarity between vector *a* and vector *b** with logistic regression to estimate the degree to which the candidate answer *b** belongs to the target class. For example, if we were trying to find the capital of a country, the target class would be *capitals*. With regard to the logistic regression parameters, the positive samples consisted of the available target words in the training set for that relation. The negative samples were simply random words pulled from the dictionary, equal to the number of positive samples. During their tests, when it came to parameters such as the number of random words chosen as negative samples or regularisation strength, no set of parameters yielded significant gains over the default choice.

In brief, LRCOS calculates the probability of a word being the correct answer in a given analogy problem through multiplying the **cosine similarity of vector *b** with vector *a*** and the **probability of the word represented by vector *b** belonging to the target class, estimated using logistic regression**.

2.4 Summary

So far, we have presented four different analogy test sets commonly used in studies testing and comparing the performance of different word vector spaces. They are the SemEval-2012 test set, the Microsoft Research test set, the Google test set, and The Bigger Analogy Test Set (BATS). The first is purely semantic, the second purely morphosyntactic, the third an uneven mix of both, and the fourth a balanced test set with both semantic and morphosyntactic categories.

Methods for solving analogy questions can be divided into pair-based methods—which learn the relation based on a single pair of words—and set-based methods—which learn the relation over a set of pairs representing it. Examples of pair-based methods are 3COSADD and 3COSMUL, and examples of set-based methods are 3COSAVG and LRCOS.

In the next section, we will thoroughly compare performance of these analogy resolution methods on different analogy tests sets. We will also study how accuracy is affected by the choice of algorithm and the parameters used to generate the word embeddings.

2.5 Results

2.5.1 Linzen (2016)

Linzen (2016) compared performance of vector spaces generated using different context window sizes on the analogy resolution task. Besides this difference in window size, all models were trained using the `word2vec` skip-gram with negative sampling (SGNS) algorithm on a concatenation of ukWaC (Baroni, Bernardini, Ferraresi, & Zanchetta, 2009) and a 2013 dump of English Wikipedia. Results show a slight advantage for 3COSMUL over 3COSADD—in line with the results of Levy et Goldberg (2014)—and very high accuracy for ONLY-B in the plurals category.

The latter can be explained by the fact that the vector of the plural form is usually one of the nearest neighbours to the vector of the singular form since they frequently appear in similar contexts. Also, Linzen states that a context window of size 10 yields better results in some world-knowledge categories: 0.68 versus 0.42 for window of size 2 in the US cities category, as shown in Figure 2.7. Smaller window sizes therefore tend to be more adapted for capturing syntactic information, and larger ones for semantic information—this is consistent with previous studies (Redington, Chater, & Finch, 1998 ; Sahlgren, 2006).

This brings us to an important point that Linzen makes: word vector spaces may seem to perform similarly with regard to average accuracy, but a closer look at performance on different categories shows us that results vary across categories for different vector spaces. This is why we must consider the downstream task when selecting model parameters—smaller context windows for formal linguistic properties, larger context windows for semantic properties.

2.5.2 Drozd et al. (2016)

In Figure 2.8, overall results show that LRCOS outperforms 3COSADD and 3COSAVG, but accuracy varies across different relations and models. That being said, 3COSAVG yields better results than LRCOS when using the skip-gram architecture. Mean_{all} is the average accuracy in the total data set, Mean_{rel} is the average score between the 14 different categories and SD is the standard deviation between categories. Usually, other studies only report the Mean_{all} value; the numbers show that Mean_{all} is generally higher than Mean_{rel}. Since the evaluation was conducted on the Google analogy test set, this sheds some light on the effect that the unbalanced nature of this test set can have on results. For a more truthful depiction of model accuracy, results should henceforth be reported per category and not over the whole test set.

As mentioned in Section 2.2.4, BATS was developed to provide a solution to the problem of unbalanced test sets. Performance on the analogy resolution task drops

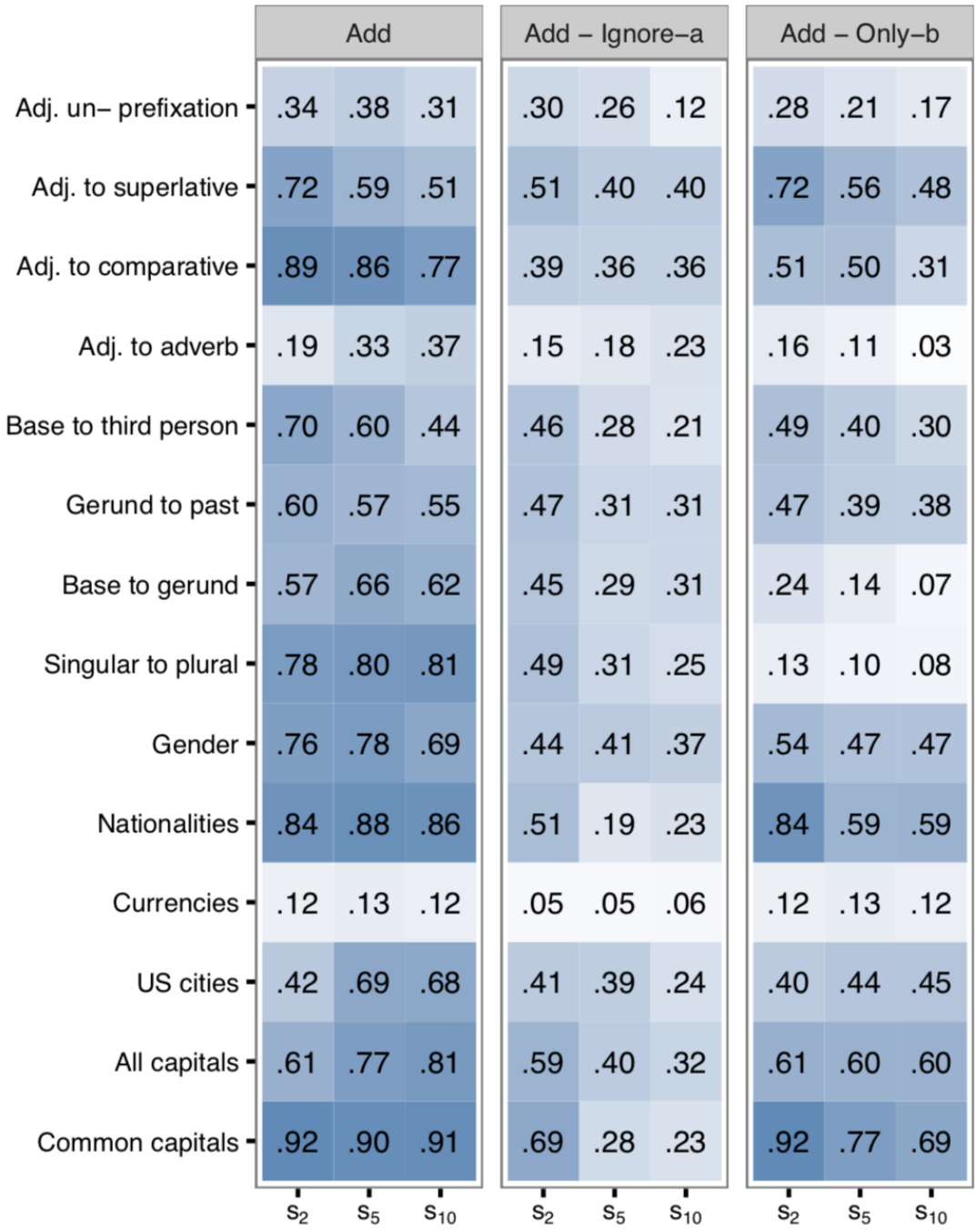


Figure 2.7 – Breakdown of results by category across different context window sizes (Linzen, 2016)

Method	3CosAdd			PairDistance			3CosAvg			LRCos		
	Mean $_{all}$	Mean $_{rel}$	SD	Mean $_{all}$	Mean $_{rel}$	SD	Mean $_{all}$	Mean $_{rel}$	SD	Mean $_{all}$	Mean $_{rel}$	SD
SVD300	50.6%	45.1%	24%	22.7%	16.1%	17%	54.8%	51.2%	26%	68.2%	68.1%	23%
SVD1000	58.1%	49.4%	25%	23.6%	22.3%	17%	59.7%	54.0%	27%	74.6%	72.6%	21%
GloVe	**79.6%**	67.8%	26%	**33.5%**	**26.9%**	22%	79.1%	74.2%	28%	73.7%	70.9%	24%
Skip-Gram	75.1%	**66.6%**	23%	28.6%	24.2%	21.6%	**80.3%**	**78.3%**	17%	**79.8%**	**78.0%**	17%

Figure 2.8 – Results on the Google analogy test set (Drozd et al., 2016)

significantly on this test set, due to low accuracy in the derivational and lexicographic categories. Figure 2.9 shows the results of the study on this test set, for 3COSADD, 3COSAVG and LRCOS on an SVD-based explicit model weighted using PPMI and a GloVe model. Both word embedding spaces were computed with a window size of 8 and dimensionality of 300.

In Figure 2.9, the count-based SVD model shows poorer performance on the inflectional morphology category than the GloVe model, which is why it gained more accuracy from LRCOS. That being said, the benefit of LRCOS was smaller on the overall better performing predictive models such as GloVe and `word2vec`, and it ***very rarely surpasses* 3COSADD *in categories where* 3COSADD *already achieved near 80% accuracy.***

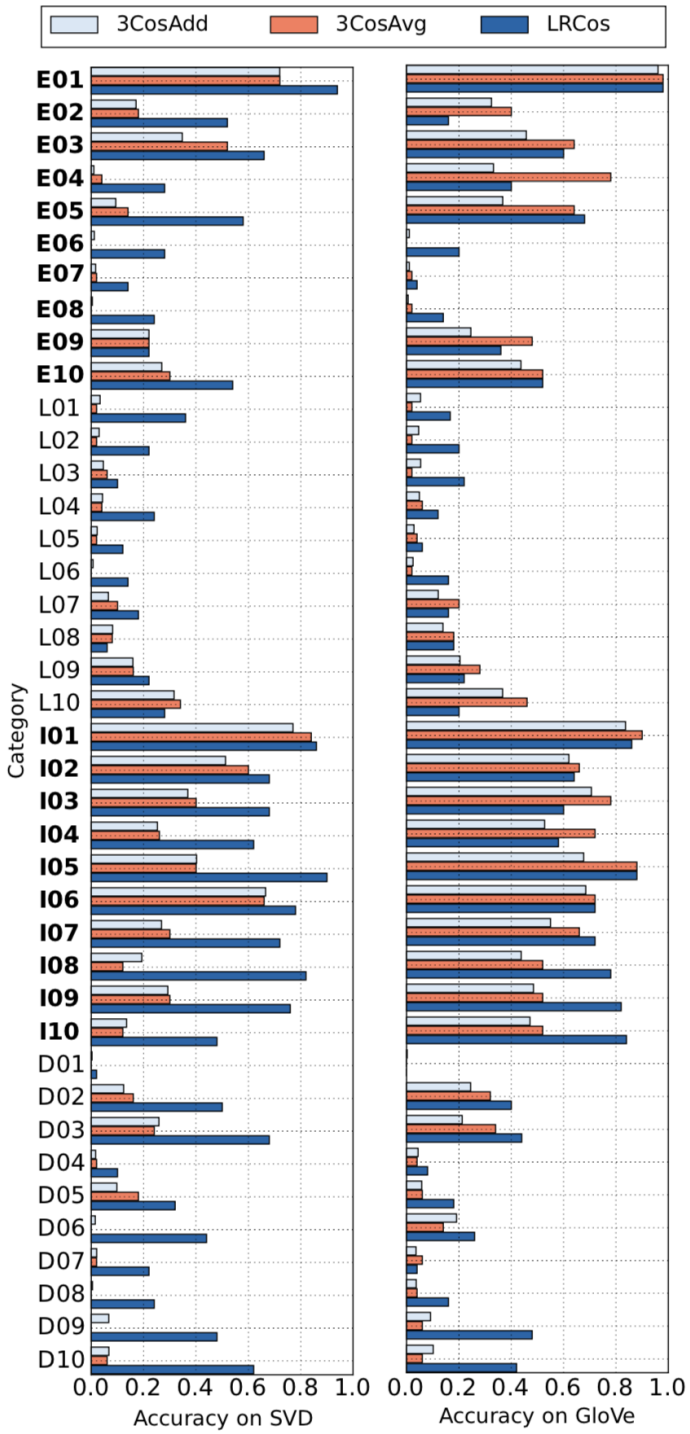
The authors also evaluated performance using models computed with dimensionality greater than 300—up to 1000 for implicit models and 1200 for explicit models—and observe that higher dimensionality does not lead to higher accuracy for the analogy resolution task. Their hypothesis is that once the core aspects of an analogical relation have been encoded in the vectors, adding more dimensions only increases noise: for most categories, increased dimensionality did not lead to improvement of results, but sometimes a slight degradation.

2.5.3 Finley et al. (2017)

This study focuses more on what the results obtained when performing the analogy resolution task on word embeddings can tell us about the compositionality of word vectors.

Their test set is composed of analogy questions aggregated from the four different analogy test sets that we presented in Section 2.2, and their best performing word vectors were generated using the `word2vec` CBOW architecture (window size 8, dimension size 200) on a 2015 dump of English Wikipedia. Both 3COSADD and 3COSMUL were tested, but only results for the former are presented due to very similar performance. As a mesure, *reciprocal rank of gain* (RGG) is used instead of accuracy, which is considered too coarse of a evaluation metric.

In the case of named entities, high RGG is shown, especially in categories dealing with capitals. On the other hand, mixed results are reported for the derivational mor-



Encyclopedic relations

E01: geography: capitals (*Athens:Greece*)
E02: geography: languages (Peru:*Spanish*)
E03: geography: UK counties (*York:Yorkshire*)
E04: people: nationality (*Lincoln:American*)
E05: people: occupation (*Lincoln:president*)
E06: animals: the young (*cat:kitten*)
E07: animals: sounds (*dog:bark*)
E08: animals: shelter (*fox:den*)
E09: thing:color (*blood:red*)
E10: male:female (*actor:actress*)

Lexicographic relations

L01: hypernyms: animals (*turtle:reptile*)
L02: hypernyms: miscellaneous (peach:*fruit*)
L03: hyponyms: miscellaneous (*color:white*)
L04: meronyms: substance (*sea:water*)
L05: meronyms: member (*player:team*)
L06: meronyms: part-whole (*car:engine*)
L07: synonyms: intensity (*cry:scream*)
L08: synonyms: exact (*sofa:couch*)
L09: antonyms: gradable (*clean:dirty*)
L10: antonyms: opposites (*up:down*)

Inflectional Morphology

I01: noun sg:pl (regular) (*student:students*)
I02: noun sg:pl (irregular) (*wife:wives*)
I03: adjective: comparative (*strong:stronger*)
I04: adjective: superlative (*strong:strongest*)
I05: infinitive: 3Ps.Sg (*follow:follows*)
I06: infinitive: participle (*follow:following*)
I07: infinitive: past (*follow:followed*)
I08: participle: 3Ps.Sg (*following:follows*)
I09: participle: past (*following:followed*)
I10: 3Ps.Sg : past (*follows:followed*)

Derivational Morphology

D01: noun+ness (home:*homeness*)
D02: un+adjective (*able:unable*)
D03: adjective+ly (*usual:usually*)
D04: over+adjective (*used:overused*)
D05: adjective+ness (*mad:madness*)
D06: re+verb (*create:recreate*)
D07: verb+able (*edit:editable*)
D08: verb+er (*bake:baker*)
D09: verb+tion (*continue:continuation*)
D10: verb+ment (*argue:argument*)

Figure 2.9 – Performance of 3COSADD, 3COSAVG and LRCOS on BATS (Drozd et al., 2016)

phology category, and ever poorer results for lexical semantics—with the exception of male-female analogies which performed rather well in this category. The authors draw from linguistic theory in an attempt to explain the variations in accuracy across analogy categories, a novel approach.

Semantics

Results are superior for categories containing named entities than other lexical categories such as common nouns or verbs. This is especially true when both the pair's words are named entities, which means that performance in the country-capitals category is especially good. This may be explained by Montague semantics, a theory of natural language semantics and of its relation with syntax (Janssen, 2017).

Montagovian set-theoretic semantics create a distinction between words of type *e* and words of type < *e, t* > in the following manner:

— **arguments of type *e***: proper nouns, denote *individuals*

— **predicates of type < *e, t* >**: verbs and common nouns, denote *sets* of individuals

The interpretation is that named entities point to a sole real-world referent: there is only one *Paris*. By contrast, a set of real-word referents exists for common nouns such as *cat*. Consequently, the contexts in which *cat* appears in are vastly more diverse than for a proper noun such as *Paris*. This may lead to more precise computation of vector representations of proper nouns.

2.6 Summary

In this chapter, we provided a thorough presentation and analysis of different analogy test sets, methods for solving analogies, and results of different analogy resolution experiments. The most important conclusion that we can draw from these results is that there is interaction between three factors: the parameters chosen to generate a vector space, the categories of word relations tested, and the methods used for solving these analogies.

For example, Drozd et al. (2016) show that the SVD model reaches state-of-the-art performance with LRCOS, which indicates that the relation information was in fact encoded in the embedding space. This attributes the poor performance of count-based models in previous studies to inappropriate choices of analogy resolution methods. It is thus possible that poor performance using one method could be improved using a different method on the same vector space.

Since the results demonstrate variation across categories, methods and vector spaces, selection of resolution methods and model parameters should be carefully based on the downstream task at hand. If several different categories of relations

are involved in the task, we may consider using separate vector spaces trained on different text data with various parameter settings. Future work could also focus on optimising methods for analogy resolution of specific categories, with respect to properties of each category and vector space. That being said, the heterogeneity of results across analogy resolution experiments suggests that the word analogy resolution task is better suited as a means of exploration of a word embedding space, rather than an evaluation.

Part II

Ideation

CHAPTER 3

PEPPER THE ROBOT

Contents

3.1 Introduction

We had the opportunity to work with a humanoid robot created by Softbank Robotics, ***Pepper***. Pepper the Robot was created with the main purpose of social interaction with humans in mind: to serve as a companion robot inside people's homes, to help the elderly, assist and watch over the young, and to entertain people. However, most Peppers in the world today are deployed in business-to-business (B2B) contexts, such as commercial stores. This is partly due to the difficulty Pepper has when conversing with a human, regardless of the age of the person or the domain of the conversation.

In this section, we will outline the current state of Pepper's dialogue engine, and discuss the effect that Pepper's humanoid form has on interaction with humans.

3.2 Pepper's Dialogue Engine

Pepper's dialogue engine is based exclusively on rule-based methods. This means that Pepper has a database of rules that map human input sequences to robot response sequences. Let us break down this process.

3.2.1 qiChat

First of all, these rules are written in a scripting language—**qiChat**—developed by the Dialog team of Softbank Robotics Europe (SBRE). qiChat was inspired by ChatScript, which was used to create the chatbot that won the 2010 Loebner Prize, an annual competition that recompenses chatbots considered to be most human-like by the judges. The rules are organised by *topic*, divided by the subject matter. For example, a topic called *Hello* or *Goodnight* would deal with salutations, whereas a topic called *Robot Capacity* would include rules about what Pepper can and cannot do. [1]

To better visualise the way qiChat works, below are two examples of rules written in qiChat:

```
#rule: hello#
u:(_[hello howdy hiya hi greetings aloha "hey there" "salutations
 $salutations=true"])
^first[
    "$greeting==done ^enableThenGoto(hello_again)"
    "$salutations==true salutations ^clear(salutations) $greeting=done"
    "^rand[$1 hello howdy hiya hi greetings aloha "hey there"]
    $greeting=done"
]

#rule: hello_again#
u:(^empty) %hello_again
[
    "Hello again"
    "You've already said hello! But I appreciate it!"
    "You really like greetings! That's nice of you!
    ^rand[hello howdy hiya hi salutations aloha "hey {there}"]"
]
```

The `u:` keyword is used before a human input pattern. Square brackets are used to hold different possibilities and will match one of the elements contained within them. They can be interpreted as an *or* operator. In the first rule: *hello* **or** *howdy* **or** *hiya* and so on.

We can create and reference variables using `$` followed by the variable name, and use them to store states and control the flow of the dialogue. In the above examples, `$greeting` is used to store information about whether or not the human has already

1. for more information on qiChat, consult the official documentation at https://android.aldebaran.com/sdk/doc/pepper-sdk/ch4_api/conversation/qichat/qichat_index.html

greeted the robot, and adapts the robot's response accordingly. Variables can also be used to force a specific robot output given a specific human input, as shown with the `$salutations` variable above.

Bookmarks, denoted by `%`, are another way of controlling the flow of dialogue. The keywords `^first` and `^rand` are used to control the way the robot chooses a response. In the former, the robot's text to speech (TTS) system outputs the first response that satisfies all conditions, whereas in the latter the robot response will be chosen at random among the given output patterns.

3.2.2 Pattern Matching

When Pepper detects speech, its automatic speech recognition (ASR) system converts the spoken language information into text. Given this input string, the dialogue engine searches for a match among the human input sequences available in Pepper's database of topics written in qiChat. If a rule is matched, the robot output sequences in that rule determine Pepper's response. This is not a fail-safe method, as the ASR result can be erroneous, or there may be no existing rule that matches the input string.

This second issue has always been a known setback of purely rule-based dialogue systems. For closed-domain dialogue systems, a rule-based approach is feasible as the domain is restricted. For instance, Pepper can be deployed by businesses because the dialogue content that needs to be created is specific to a predefined use case, which makes the task of creating rules for this particular scenario possible. However, for an open-domain dialogue system—which Pepper, being a social robot, should be—a purely rule-based approach is impossible. No number of writers or automatisation would ever be able to accomplish this task, as a defining feature at the core of human language is its infinite number of possible combinations.

3.2.3 Knowledge

When it comes to storing and accessing information, Pepper is equipped with a triple-based knowledge representation system. In practice, its main purpose is to allow the robot to store information about the user, such as their name and certain preferences. In theory, we could enrich this knowledge base with any information in triple form and create an inference system in order for the robot to be able to exploit this information during dialogue. This is nonetheless a rather static approach, in that the knowledge base would have to be engineered by a human expert and would eventually become too costly to maintain and develop manually.

3.3 Human-Robot Interaction

A particularity that Pepper possesses which differentiates it from other dialogue systems is obviously its humanoid form. This changes the circumstances of interaction, most importantly by creating expectations for the robot's behaviour. Humans interacting with a humanoid robot tend to project expectations for the robot to be the most human-like possible. This means that a human interacting with Pepper, especially for the first time, may have a tendency to speak to it exactly as they would speak to another human. Alas, a robot does not understand human language.

Let us use an example to illustrate the effect of these expectations. The ASR result is the first step, or obstacle, in human-robot interaction. To favour success, the human must articulate well and speak at a moderate pace. Furthermore, if the human makes mistakes, false starts, uses filler words, or if their speech contains any other disfluency, this will create complications. These speech disfluencies are an intrinsic part of spontaneously uttered speech (Corley & Stewart, 2008 ; Lickley & Bard, 1998). It is therefore not natural to erase these phenomena from our speech during verbal interaction with a robot. That being said, perhaps human-robot interaction will one day lead to the evolution of a new language form specific to this type of interaction, brought forth by the rising necessity for humans to communicate efficiently with machines.

3.4 Summary

In this third chapter, we provided a brief introduction to Pepper the Robot, as well as its dialogue and knowledge engines. We have seen that its dialogue engine is a purely rule-based one, and that its knowledge engine is essentially a knowledge base containing triples that we can manipulate when creating dialogue content. Both of these approaches are primarily suited to closed-domain dialogue systems, as they are too costly or even impossible to develop and maintain on a regular basis for an open-domain dialogue system.

Herein lies the inspiration for the experiment conducted in this study.

CHAPTER 4

BOUNCING BACK WITH WORD EMBEDDINGS

Contents

4.1 Introduction

As seen in Part 1, word embeddings generated by neural networks do indeed seem to be capable of encoding linguistic regularities within their word vectors. The question we aim to explore by conducting this experiment is the following: ***to what extent can we exploit the characteristic of neural word embeddings being able to capture semantic and morphosyntactic regularities as a way to generate responses in a dialogue system?*** Additionally, what are some potential advantages and disadvantages of such an approach?

Before investigating the second question, let us explain how the idea in the first question came about. The initial hurdle we were trying to overcome was being able to spontaneously generate a response in cases of unmatched input sequences. Our core inspiration comes from verbal interaction situations in which one of the interlocutors does not have something to say, but merely wishes to provide a response so as to keep the interaction going. Through simulation scenarios and the analysis of dialogue corpora, we concluded that humans tend to fall back on elements from previous utterances in order to produce a novel utterance. In other words:

1. we can select certain parts—such as a noun phrase—from the previous utterance, and produce a new utterance pertaining to or containing these parts

2. we can select certain parts and then use mental association to come up with a new phrase around which we construct our new utterance

We call this mechanism "*bouncing*", as we are using the previous utterance as a trampoline to our new utterance. In more formal terms, the *parts* that we select in the previous utterance are generally noun phrases, and *mental association* refers to some type of relation that links the first noun phrase to the second, most likely a semantic one.

```
speaker1:  croissants in france are so yummy right
speaker2:  yeah i mean their pastries are heavenly
speaker1:  true i ate way too much brioche this weekend
speaker2:  i drank too much ha ha
```

In the above example, we spot semantic links between two groups of words, the words in orange and the words in purple. Concerning the words in orange, we go from *croissants* to *pastries*, its hyperonym. We then go from the hypernym back to another hyponym, *brioche*. This relation of hypernym and co-hyponyms creates a meaning relation between the utterances, and establishes a sense of relevance throughout the interaction. Thus, examining these relations may be able to provide us with a phrase-generating mechanism based on bouncing off an element of the previous utterance in order to produce a new utterance that would remain pertinent in the context of the dialogue.

So, seeing as neural word embeddings encode semantic and morphosyntactic information within their vectors, could we use these word embeddings as a base for concept association based on semantic relationships?

4.2 Potential Advantages

The most striking benefit of using such a tool is the fact that neural network models such as `word2vec` can be trained on unannotated text data. Reducing the need for manual or semi-manual annotation would therefore be favourable, as annotation can be a relatively costly task in a data pipeline containing natural language processing.

That being said, simply because no human annotation is required does not mean that `word2vec` is a truly unsupervised machine learning algorithm. Neural networks work by backpropagating error, and calculating error requires labelled data. `word2vec` includes a preprocessing step that creates its own labels and then trains the network and generates embeddings. Consider the following example:

```
the cat in the hat

-- (the, cat)
-- (cat, the), (cat, in)
-- (in, cat), (in, the)
-- (the, in) (the, hat)
-- (hat, the)
```

Given the input text, the network makes data pairs based on the window size. Here, the window size is 1, so for each word the network considers one word from history and one from future. Thus it creates labels and trains the network. This is a particular instance of supervised learning called *semi-supervised learning*, in which the targets are generated from the input data. No human expert is needed for making the labels.

Word embeddings can be said to be dynamic, as training the same model on different corpora would produce different results. This is to say that the compositionality of the computed word vectors would not be identical if the same model was separately trained on a Wikipedia corpus and on a literary corpus. The choice of corpus would then depend on specifics of the downstream task at hand—in this case, the purpose of the dialogue system, its domain, or even its personality.

Finally, it has been shown that word embeddings encode information about linguistic relations in their word vectors. This is a helpful trait for our experiment as our aim is to use mainly semantic relations to bounce from one concept to another, and then build a phrase around that concept. In theory, a structured lexicon created by humans, such as WordNet (Miller, 1995), would fit the goal of our dialogue system. However, experimentality is at the core of this work, hence the choice of using of a semi-supervised machine learning algorithm to create this concept association mechanism in our dialogue system.

4.3 Potential Disadvantages

The fact that `word2vec` is a semi-supervised learning algorithm is a double-edged sword. On one hand, it allows us to do without the need of human annotation of the training data, as the neural network creates its own labels through backpropagation. On the other hand, this also implies that we relinquish control and the ability to ensure the generated word embeddings follow a certain structure or contain specific information. Faruqui et al. (2014) show that it is indeed possible to retrofit word vectors with information from semantic lexicons, and this could perhaps be explored in the future for a domain-specific dialogue system. For now, the aim of this study is

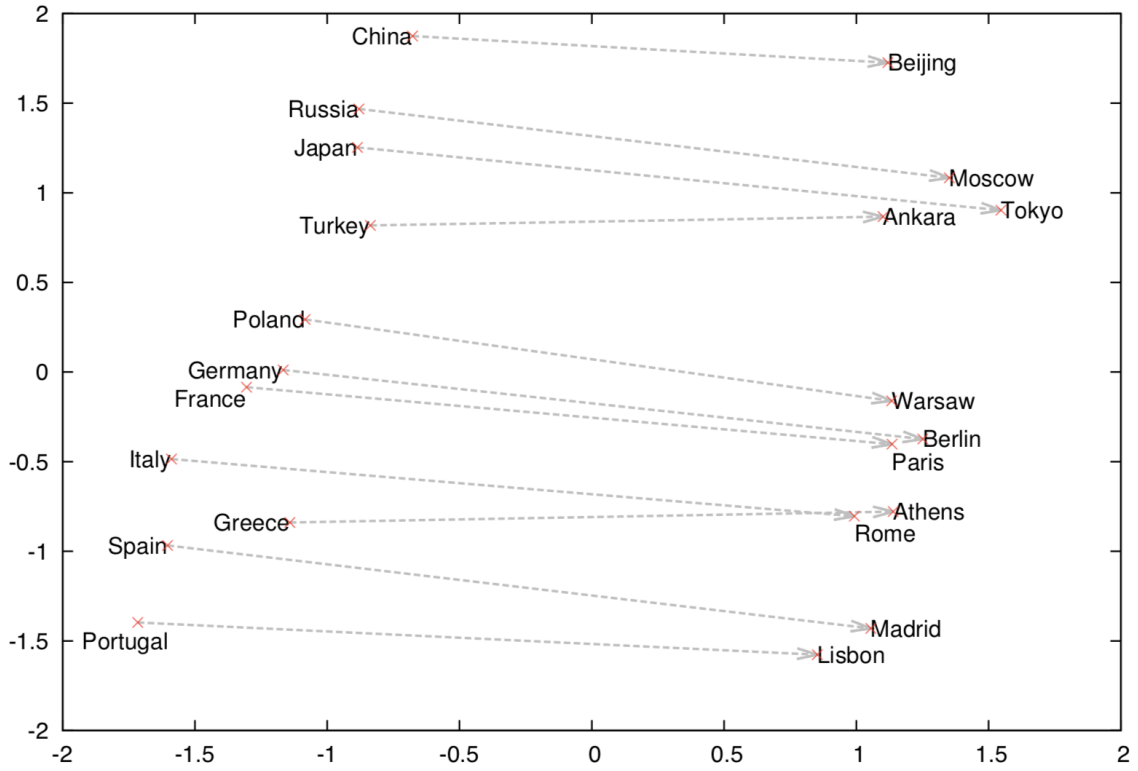


Figure 4.1 – Two-dimensional PCA projection of countries and their capital cities (Mikolov, Sutskever, et al., 2013)

to conduct this initial experiment and see how word embeddings can be used as a cog of a natural language generation mechanism in a dialogue system.

This also means that since the creation of the word embeddings is a semi-supervised task, the composition of the word vectors remains somewhat of a mystery to us, at least at the beginning. We could begin to familiarise ourselves with the compositionality of a word vector space using visualisation methods such as principal component analysis (PCA) (Pearson, 1901) and t-Distributed Stochastic Neighbour Embedding (t-SNE) (van der Maaten & Hinton, 2008). In Figure 4.1, PCA is used to visualise skip-gram vectors with 1000 dimensions in a two-dimensional space. We observe the constant relation offset between countries and their capital cities, exhibiting how the model implicitly learns relationships between concepts after automatically organising them during training. No information about what a capital city means was provided to the model during training.

Since both PCA and t-SNE are based on dimensionality reduction, the tradeoff when employing these methods is that we lose a significant amount of the information generated by the neural network. We recommend coupling such visualisation methods with approaches similar to Finley et al. (2017), based on drawing from linguistics to explain variations in performance across categories of semantic, syntactic, and morphological properties on the analogy resolution task.

To conclude, as the Part 1 has shown us, word embeddings are far from achieving perfect accuracy on all the word analogy categories, which would be an obvious issue for dialogue systems in production. That being said, word embeddings yield consistently good results on certain categories such as the country-capital one and the singular-plural one. As a start, we could focus on these categories in order to ensure good overall accuracy of the systems, but this is rather restrictive, and results still do not show perfect accuracy.

4.4 Summary

In this section, we presented our hypothesis for why word embeddings could be a way to recreate, in a dialogue system, the concept association mechanism in human verbal interaction. We also went through the potential advantages and disadvantages of this method. In the next section, we will present the steps we undertook in the creation of our dialogue system.

Part III

Implementation

CHAPTER 5

DIALOGUE SYSTEM

Contents

5.1 Introduction

This chapter will present the different components of our generative dialogue system and the steps taken to build them. The first step was the creation of a basic dialogue system inside of which we integrate our response mechanism. Next, we present the corpus and model we used to compute our word embeddings. Finally, we will go through how we generated responses using Markov chains.

5.2 A Simple Dialogue System

First of all, we need some kind of structure to house our experiment—a skeleton, if you will. For this purpose, we create a simple interface using Python. When we enter a human input, the first step is to process this input by means of natural language understanding techniques. We use the **spaCy** NLP library for this.

Upon receiving a valid human input, we annotate the text data with linguistic information. The tags that will primarily be useful to us in this study are the part-of-speech (POS) tags. We also perform a dependency parse of the input sentence.

We extract tokens from the human input that are either common or proper nouns, denoted respectively by the POS tags `NOUN` and `PROPN`. If a noun shares a dependency with another element of the sentence, such as being a prepositional object, we extract the other noun in the relation. We sometimes extract the root verb of the human input sentence.

The reason we extract these elements from the human input is so they may serve in the next step, during which we access a word embedding model and search for the most similar terms given one or several words. Other inputs that may fall into broad categories, such as the *greetings* category, are dealt with using pattern-matching rules that we created for our dialogue system. This means that our system is a **hybrid** one, as it employs both retrieval-based methods and generative-based methods in providing a response.

5.3 Word Vector Space

One constraint that we note at this point is that the words must exist in the vocabulary of our word embedding model for there to be a match. Consequently, since our dialogue system is open-domain, it would be beneficial to use word embeddings generated by training on a rather large data set of generalist text. Also, in order to increase accuracy, higher dimensionality is recommended. We will address these issues in this section.

5.3.1 Corpus

Reddit is an American discussion website founded in June 2005 by Steve Huffman and Alexis Ohanian. It can be described as a hybrid social network and forum for sharing and discussing news and web content, and asking and answering questions on a vast variety of subjects. These topics are organised by boards called *subreddits*, created and maintained by the users, called *redditors*. In February 2018, Reddit occupied 3rd position in a ranking of most viewed websites in the United States, and 6th position worldwide, as reported by Alexa Internet, a web traffic analysis company. [1]

Here are some statistics on Reddit in 2015, taken from the official Reddit blog [2]:

- 82.54 billion pageviews
- 73.15 million submissions
- 725.85 million comments:
  - made by 8.7 million total authors
  - containing 19.36 billion words

1. https://www.alexa.com/siteinfo/reddit.com
2. https://redditblog.com/2015/12/31/reddit-in-2015/

— 6.89 billion upvotes
— 88,700 active subreddits

The wealth of text data and large number of subjects discussed on Reddit suit our needs perfectly. Qualitatively, we find the overall language and writing style of redditors to be fitting; spelling, grammar and punctuation are generally of acceptable quality, and the style is casual and usually good-spirited, even when debating. This is why we decided to use a corpus of Reddit comments posted in January 2015 to train our word embedding model. It is a subcorpus of a huge data set [3] comprised of all publicly available Reddit comments, collected and made available by Jason Baumgartner, who maintains a website dedicated to learning about big data and social media analysis [4].

The comments were initially in JSON format and included information such as author, subreddit, comment score, and comment body. We extracted only the comment body text from the JSON objects and proceeded to preprocess the text before feeding it into a neural network for generating word vectors. We used regular expressions to delete URL links and insert line breaks after each sentence. We also removed HTML tags, converted all letters to lowercase and numbers to their alphabetical counterparts, and inserted whitespace between words and punctuation marks. After this preprocessing step, our corpus consists of:

— 811,388,734 tokens
— 56,099,311 lines

To better visualise this, here are samples of our text data before preprocessing:

```
What an interesting question!  I would have to say...  We don't
know.  We know there are two pigments essentially a brown and
red pigment that make up hair color.  I'm pretty sure that
the amount of red is controlled by 1 gene (but MANY different
alleles.)  We are not sure how many genes are involved in the
amount of brown pigment.  We just don't know enough to be able
to predict offspring hair color.
```

Comments can be specific to a domain, or purely anecdotal:

```
...Drink it.  Hahaha, that's awesome.  I've done something
similar - saw what I thought was a stooped, older gentleman on
the bus (his head was turned to the side, face partially hidden
by scraggly grey hair and a wool cap, and the skin on his hands
```

3. for the full corpus of 1.7 billion Reddit comments visit:
https://www.reddit.com/r/datasets/comments/3bxlg7/i_have_every_publicly_available_reddit_comment/
4. https://pushshift.io/

```
looked pretty rough) and stood up to offer my seat.  He tried
to decline (head still turned away) but I insisted.  He finally
took the seat in resignation, looked up at me as he sat down
and muttered "I'm not even that old".  He was probably in his
early-mid fourties.
```

After preprocessing, our text data looks like this:

```
what an interesting question !
i would have to say ...
we don't know .
we know there are two pigments essentially a brown and red
pigment that make up hair color .
i'm pretty sure that the amount of red is controlled by one gene
( but many different alleles .)
we are not sure how many genes are involved in the amount of
brown pigment .
we just don't know enough to be able to predict offspring hair
color .
...  drink it .
hahaha , that's awesome .
i've done something similar - saw what i thought was a stooped
, older gentleman on the bus ( his head was turned to the side
, face partially hidden by scraggly grey hair and a wool cap ,
and the skin on his hands looked pretty rough ) and stood up to
offer my seat .
he tried to decline ( head still turned away ) but i insisted .
he finally took the seat in resignation , looked up at me as he
sat down and muttered " i'm not even that old " .
he was probably in his early-mid fourties .
```

5.3.2 Gensim and `word2vec`

We used the **Gensim** Python implementation of the `word2vec` algorithm. Our word vectors were computed with the following parameters:

- **minimum count**: 10 (only words appearing at least ten times in the corpus are considered)
- **vector dimension size**: 300
- **context window size**: 5
- **training algorithm**: skip-gram

— **negative samples**: 10

After 5 training epochs completed in around an hour and a half on our corpus of over 800M tokens, our word vector space is generated with 300 dimensions and a vocabulary size of 751,111 words.

5.3.3 Vector Space Exploration

Let us see if our word embeddings are able to solve some of the traditionally used analogy examples:

REDDIT WORD VECTORS	
analogy	**3COSADD**
man:king::woman:?	queen, 0.6220933198928833
france:paris::germany:?	berlin, 0.6995094418525696
oregon:portland::tucson:?	arizona, 0.6880635023117065
germany:german::france:?	french, 0.8510904312133789
germany:german::netherlands:?	dutch, 0.7724204063415527
dancing:danced::jumping:?	jumped, 0.7310522794723511
dancing:danced::flying:?	flew, 0.9726194143295288
banana:bananas::bird:?	birds, 0.669019341468811
bottle:bottles::car:?	cars, 0.8225154876708984
japan:yen::europe:?	euros, 0.6935263872146606

Table 5.1 – Results using 3COSADD to solve analogies in our vector space

REDDIT WORD VECTORS	
analogy	**3COSMUL**
man:king::woman:?	queen, 0.8622549772262573
france:paris::germany:?	berlin, 0.8249755501747131
oregon:portland::tucson:?	arizona, 0.8815611004829407
germany:german::france:?	french, 0.970018744468689
germany:german::netherlands:?	dutch, 0.9512843489646912
dancing:danced::jumping:?	jumped, 0.980408787727356
dancing:danced::flying:?	flew, 0.6888895034790039
banana:bananas::bird:?	birds, 1.0101929903030396
bottle:bottles::car:?	cars, 1.0429044961929321
japan:yen::europe:?	euros, 0.8800071477890015

Table 5.2 – Results using 3COSMUL to solve analogies in our vector space

Tables 5.1 and 5.2 demonstrate our vector space's ability to represent these relations among its word vectors. Table 5.1 shows scores when using the 3COSADD method for solving analogies, and Table 5.2 shows scores for 3COSMUL. Both methods are successful in retrieving the correct response, but 3COSMUL is more accurate, which is why we will be employing this method in our dialogue system.

We shall explore our word embeddings further by looking at the vectors closest to some words:

REDDIT WORD VECTORS	
word	**NEIGBOURHOOD**
pizza	burger, 0.8053134083747864 sandwich, 0.7730149626731873 burrito, 0.7347080707550049 sushi, 0.720064103603363 chipotle, 0.7158690690994263 steak, 0.7064655423164368 cheeseburger, 0.7061429023742676 kfc, 0.7012215852737427 cheesesteak, 0.696954309940338 1 hotdog, 0.6860557794570923

Table 5.3 – Vectors closest to *pizza* in our vector space

REDDIT WORD VECTORS	
word	**NEIGBOURHOOD**
pants	trousers, 0.7896082401275635 undies, 0.7825345993041992 jeans, 0.7795000672340393 socks, 0.7662390470504761 shorts, 0.7512415051460266 underwear, 0.7492477893829346 sweatpants, 0.7492419481277466 boxers, 0.7377183437347412 shoes, 0.7275022864341736 shirt, 0.7163687944412231

Table 5.4 – Vectors closest to *pants* in our vector space

The results in Tables 5.3 and 5.4 are an indication of our model's capacity to group together concepts belonging to the same semantic class. In effect, when exploring the respective neighbourhood structures of *pizza* and *pants*, we observe that the terms nearest to them are co-hyponyms of the same class, *(fast) food* for the former and *clothing* for the latter.

Now for something more fun. What happens when we add the vectors of *cat* and *cute*? *Red* and *purple*? *Puppy* and *adult*?

```
cat + cute = kitten, 0.7406163215637207
red + blue = purple, 0.7893121242523193
puppy + adult = toddler, 0.6859981417656 ; dog, 0.6807955503464
```

This last example is especially interesting. By adding the vectors of *puppy* and *adult*,

the top result was the vector for *toddler*, which may mean that our model somehow encoded the *baby* property in the vector for *puppy* and the *human* property in the vector for *adult*. This is a possible explanation for why adding these two vectors resulted in the vector closest to *toddler*, a human baby.

Now that we have generated our word embeddings, the words that we extracted from the human input in the first step of the process are used as elements to extract new, related terms from our word vector space.

5.4 Response Generation

In the previous step, the result was a list of words extracted from the vocabulary of our word embedding model. Given this list of words, we aim to generate a response to the human input. To do so, we have decided to implement a Markov chain text generator using n-grams.

5.4.1 *n*-grams

The term *n*-gram is used to denote any sequence of *n* elements, where *n* is called the *order* of the *n*-gram. The elements in the sequence can be letters, phonemes, syllables; in our case they are words. *n*-grams can be used to build a statistical model of text. This will help us assemble the *n*-grams into a Markov chain, explained in the next section. We will be using 2-grams—or **bigrams**—illustrated below.

```
the cat in the hat

-- (the, cat)
-- (cat, in)
-- (in, the)
-- (the, hat)
```

5.4.2 Markov Text

First of all, a Markov process, named after the Russian mathematician Andrey Markov, is one that exhibits the Markov property of memorylessness. If prediction for the next step of the process can be made based solely on the state of the current step, then the process is said to satisfy the Markov property. (Gagniuc, 2017) A Markov chain is hence a model that describes a sequence of events in which each step depends probabilistically only on the current state.

A Markov chain can be used on the n-grams collected from a text corpus as a way to generate new sequences of text. The first step of this process is to select text data and build an n-gram dictionary. This means that for each n-gram, we record the n-gram that follows it and create a dictionary storing this information. To better visualise this process, imagine a tree where each word is represented by a branch. The leaves on each branch are words that followed the branch word somewhere in our corpus. The size of the leaf words then vary depending on the number of times they co-occurred with our branch word.

We trained our Markov chain text generator on subsets of the 811M word Reddit corpus. To generate a new sequence based on our collection of bigrams, we select a start word and choose randomly among its list of subsequent bigrams, and so on, until the maximum sentence length or an end word is reached. The results in the next section were generated with a maximum sentence length parameter of 15. The start words are the words retrieved from our vector space.

5.5 Results

Here are some examples of input and output sequences produced during interaction with our dialogue system, named *chatterbox*:

human: we got **pizza** and ate in the yard
chatterbox: **pizza** boxes fell on that bench?
human: no we **ate** the pizzas
chatterbox: i **drank** all 3 in body weight and focus on the chin
human: i would rather **focus** on the **thighs**
chatterbox: **concentrate** more on building relationships. **concentrate** more on that asap.
human: thanks for the advice i guess
human: we had a really rough **night**
chatterbox: **morning** honey, how'd you guess?
human: there were protests all over **paris**
chatterbox: **paris** catacombs left a night time shelter and protect yourself
human: i stayed inside, i don't think people were **hiding** in the **catacombs**
chatterbox: **caves** don't understand customer service.
human: do you mean they're not that **comfortable**?
chatterbox: **uncomfortable** topic. **uncomfortable** amount of hate and harm.
human: i understand, i don't really like to **discuss politics**

```
either
chatterbox:  debate is encouraged for you as well!
human:  maybe later
chatterbox:  perhaps some variant of "the master said this:
blah blah".
human:  yeah we really are like slaves
chatterbox:  slavery in 1981.  still 19 years ago?
human:  woah i didn't realize you were a social justice warrior
chatterbox:  activism say, for example, germany.
human:  what does germany have to do with anything?
chatterbox:  europe their attitudes and techniques varied
drastically based on a shoestring budget.
human:  ha ha i don't know.  can we talk about fashion instead?
chatterbox:  fashion trends and the women you pass.
human:  oh?  do you follow any bloggers?
chatterbox:  gurus are lacking beauty vids.
```

In the above excerpt, the words in boldface are the ones that were extracted from the human input and the ones retrieved from the vector space using the human input words. Despite some nonsensical responses and ungrammaticality, a few of the generated responses do exhibit a degree of coherence within the dialogue. Amusingly, we notice that the context and interpretation of the generated responses somehow create meaning for seemingly unrelated responses. Meaning is constructed throughout the dialogue with the input and output sequences, despite the system sometimes giving bizarre responses.

5.6 Summary

In this chapter, we described the steps taken in our creation process. First, we made a basic dialogue system that extracts elements from the human input. We then collected and preprocessed an 811M word corpus of Reddit comments. This corpus was used to train a `word2vec` skip-gram model with 300 dimensions and a context window size of 5. Exploration of our vector space allowed us to observe some interesting properties contained within the compositionality of our word vectors. Then, given the list of words taken from the human input, we used one or more words to extract related terms from our vector space. Finally, to generate a response to the human, we trained a Markov chain text generator on subsets of the Reddit corpus and fed it the result word from our vector algebra in the previous step as the start word of the generated sentence.

CHAPTER 6

DISCUSSION

Contents

6.1 Introduction

In this final chapter, we will reflect on our experimentation process and propose extensions and new lines of study for future work.

6.2 Discussion and Future Work

The biggest edge that using word embeddings gives us is the ability to train effectively on large quantities of unlabelled data. No lexicon or human annotation was provided, yet our model was able to encode semantic and morphosyntactic properties in its vectors. We were seeking these semantic classes in order to create a concept association mechanism in our dialogue system and be able to spontaneously generate responses based on these associations.

An ontology or a knowledge base organises words or concepts based on their denotation, which is viewed as a stable, invariable notion. However, the meaning of a word is mutable and varying; it is built by the contextualisation of lexical units in a text. This brings us back to the Distributional Hypothesis (Harris, 1954):

"linguistic terms with similar distributions have similar meanings"

The underlying idea was popularised by Firth (1957):

"you shall know a word by the company it keeps"

Using different text data and model hyperparameters when training a neural network leads to the generation of different embedding spaces. This raises the question

of transferability of our system: how would it function with word embeddings generated using different corpora and model settings? We may perhaps consider a corpus to be a representation of Pepper's semantic experience: it is Pepper's unorganised, experiential and phenomenological knowledge—an archive of its experiences. This highlights the importance of the corpus as it:

— becomes a reservoir of knowledge and experiences of the world
— may have a decisive influence on the *personality* of the dialogue system

To test this hypothesis, we could conduct the same experiment with different corpora in order to observe the relationship between the genre of the corpus and the personality of the dialogue system. Would a corpus of poetry create a poetic bot? Would a corpus of lampoons create an agressive bot?

Regarding our Markov text generator, the generated sentences were generally pleasant and somewhat well-formed. The Reddit corpus makes for an original language style. However, the responses it generates are not always completely grammatical, which would evidently pose an issue for dialogue systems in production. Nonetheless, this is something that future work could focus on and improve, perhaps through the integration of a phrase grammar. Also, the start word of our generated sentence is chosen from the related word vectors of words extracted from the human input. In order for the response to be generated successfully, the start word must be present in the corpus used to train the Markov text model.

To increase chances of success, we trained the text generator on a subcorpus of the corpus used to build our vector space. We also extracted a list of related word vectors from our embedding space instead of a single one, and fed them into the Markov text model one after the other until we were able to generate a sentence. It is still possible that none of the extracted terms allow us to generate a sentence. Furthermore, in our current model, the extracted term is always placed at the beginning of the generated sentence. Perhaps we could extend our model in the future to include a backwards n-gram text generator, where the input word is used as the end word. Finally, our Markov text generator could be enhanced through the addition of a fitness function that would allow us to modify features of the generated response. For example, we could use a fitness function to reward more complex words if we wish to give them a higher chance of appearing in the generated text, or, inversely, reward simpler words in order to use more of them.

When considering other characteristics of our system, we currently use the 3COSMUL method to extract terms from our vector space when more than one element is extracted from the human input, and return a list of nearest neighbours when a single word was extracted from the human input. It may be interesting to experiment

using different methods depending on the category of the extracted word. This can be studied alongside a more in-depth exploration of the vector space in order to gain a better understanding of its composition and the manner in which it encodes different linguistic properties.

6.3 Summary

To summarise, we could investigate the usage of different methods for extracting new terms from our vector space, based on the category of the words extracted from the human input, for instance. We could also study the effect that using different word vector spaces, generated using corpora from different genres, would have on our dialogue system. The Markov chain text generator could be improved with a grammar, a fitness function and a backwards n-gram generator. It may also be interesting to train the embedding model and the text generator on different corpora and examine the results. Lastly, robustness of the model could be increased by enabling it to better handle out-of-vocabulary words.

Conclusion

The aim of this experiment was to introduce a concept association mechanism that allows a dialogue system to generate novel responses semantically related to the input it received. This was implemented using neural word embeddings, which have shown potential by implicitly encoding linguistic properties, both semantic and morphosyntactic, in the composition of their word vectors. The internal structure of vector spaces remains relatively obscure to the scientific community. As long as this is the case, we are unable to fully exploit this powerful tool. We encourage further study of the compositionality of vector spaces, sensitive to interplay between three main factors: the text data set, the model parameters, and the methods used for information extraction.

BIBLIOGRAPHY

Baroni, M., Bernardini, S., Ferraresi, A., & Zanchetta, E. (2009, 01 septembre). The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, *43*(3), 209–226. Consulté sur https://doi.org/10.1007/s10579-009-9081-4 doi: 10.1007/s10579-009-9081-4

Baroni, M., Dinu, G., & Kruszewski, G. (2014). Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 238–247). Association for Computational Linguistics. Consulté sur http://aclweb.org/anthology/P14-1023 doi: 10.3115/v1/P14-1023

Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003, mars). A neural probabilistic language model. *J. Mach. Learn. Res.*, *3*, 1137–1155. Consulté sur http://dl.acm.org/citation.cfm?id=944919.944966

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). Enriching word vectors with subword information. *CoRR*, *abs/1607.04606*. Consulté sur http://arxiv.org/abs/1607.04606

Brants, T., Popat, A. C., Xu, P., Och, F. J., & Dean, J. (2007). Large language models in machine translation. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (emnlp-conll)*. Consulté sur http://aclweb.org/anthology/D07-1090

Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on machine learning* (pp. 160–167). New York, NY, USA : ACM. Consulté sur http://doi.acm.org/10.1145/1390156.1390177 doi: 10.1145/1390156.1390177

Corley, M., & Stewart, O. W. (2008). Hesitation disfluencies in spontaneous speech: The meaning of um. *Language and Linguistics Compass*, *2*(4), 589–602. doi: 10.1111/j.1749-818X.2008.00068.x

Drozd, A., Gladkova, A., & Matsuoka, S. (2016). Word embeddings, analogies, and machine learning: Beyond king - man + woman = queen. In *Coling 2016, 26th international conference on computational linguistics, proceedings of the confer-*

ence: Technical papers, december 11-16, 2016, osaka, japan (pp. 3519–3530). Consulté sur http://aclweb.org/anthology/C/C16/C16-1332.pdf

Dubin, D. (2004). The most influential paper gerard salton never wrote. *Library Trends*, *52*(4), 748-764. Consulté sur http://dblp.uni-trier.de/db/journals/libt/libt52.html#Dubin04

Evert, S. (2005). *The statistics of word cooccurrences: Word pairs and collocations*. Consulté sur https://books.google.fr/books?id=Uof3tgAACAAJ

Faruqui, M., Dodge, J., Jauhar, S. K., Dyer, C., Hovy, E. H., & Smith, N. A. (2014). Retrofitting word vectors to semantic lexicons. *CoRR*, *abs/1411.4166*. Consulté sur http://dblp.uni-trier.de/db/journals/corr/corr1411.html#FaruquiDJDHS14

Finley, G., Farmer, S., & Pakhomov, S. (2017). What analogies reveal about word vectors and their compositionality. In *Proceedings of the 6th joint conference on lexical and computational semantics (*sem 2017)* (pp. 1–11). Association for Computational Linguistics. Consulté sur http://aclweb.org/anthology/S17-1001 doi: 10.18653/v1/S17-1001

Firth, J. R. (1957). A synopsis of linguistic theory. , *1952-59*, 1-32.

Gagniuc, P. A. (2017). *Markov chains: From theory to implementation and experimentation* (First Edition éd.). John Wiley & Sons.

G.E. Hinton, D. R., J.L. McClelland. (1986). Parallel distributed processing: Explorations in the microstructure of cognition. In (Vol. Volume 1: Foundations, chap. Distributed representations). MIT Press.

Gladkova, A., Drozd, A., & Matsuoka, S. (2016). Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. In *Proceedings of the student research workshop, srw@hlt-naacl 2016, the 2016 conference of the north american chapter of the association for computational linguistics: Human language technologies, san diego california, usa, june 12-17, 2016* (pp. 8–15). Consulté sur http://aclweb.org/anthology/N/N16/N16-2002.pdf

Golub, G. H., & Van Loan, C. F. (1996). *Matrix computations (3rd ed.)*. Baltimore, MD, USA : Johns Hopkins University Press.

Harris, Z. (1954). Distributional structure. *Word*, *10*(23), 146–162.

Janssen, T. M. V. (2017). Montague semantics. In E. N. Zalta (Ed.), *The stanford encyclopedia of philosophy* (Spring 2017 éd.). Metaphysics Research Lab, Stanford University. https://plato.stanford.edu/archives/spr2017/entries/montague-semantics/.

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, *28*(1), 11-20.

Jurgens, D. A., Turney, P. D., Mohammad, S. M., & Holyoak, K. J. (2012). Semeval-

2012 task 2: Measuring degrees of relational similarity. In *Proceedings of the first joint conference on lexical and computational semantics - volume 1: Proceedings of the main conference and the shared task, and volume 2: Proceedings of the sixth international workshop on semantic evaluation* (pp. 356–364). Stroudsburg, PA, USA : Association for Computational Linguistics. Consulté sur http://dl.acm.org/citation.cfm?id=2387636.2387693

Lee, D. D., & Seung, H. S. (2000). Algorithms for non-negative matrix factorization. In *Proceedings of the 13th international conference on neural information processing systems* (pp. 535–541). Cambridge, MA, USA : MIT Press. Consulté sur http://dl.acm.org/citation.cfm?id=3008751.3008829

Levy, O., & Goldberg, Y. (2014). Linguistic regularities in sparse and explicit word representations. In R. Morante & W. tau Yih (Eds.), *Conll* (p. 171-180). ACL. Consulté sur http://dblp.uni-trier.de/db/conf/conll/conll2014.html#LevyG14

Levy, O., Goldberg, Y., & Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings. *TACL*, *3*, 211–225.

Lickley, R. J., & Bard, E. G. (1998). When can listeners detect disfluency in spontaneous speech? *Language and Speech*, *41*(2), 203-226. Consulté sur https://doi.org/10.1177/002383099804100204 (PMID: 10194877) doi: 10.1177/002383099804100204

Linzen, T. (2016). Issues in evaluating semantic spaces using word analogies. *CoRR*, *abs/1606.07736*. Consulté sur http://dblp.uni-trier.de/db/journals/corr/corr1606.html#Linzen16

Luhn, H. P. (1957, octobre). A statistical approach to mechanized encoding and searching of literary information. *IBM J. Res. Dev.*, *1*(4), 309–317. Consulté sur http://dx.doi.org/10.1147/rd.14.0309 doi: 10.1147/rd.14.0309

Mikolov, T. (2007). *Language modeling for speech recognition in czech* (Mémoire de Master non publié). Brno University of Technology.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, *abs/1301.3781*. Consulté sur http://arxiv.org/abs/1301.3781

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., & Joulin, A. (2018). Advances in pre-training distributed word representations. In *Proceedings of the international conference on language resources and evaluation (lrec 2018)*.

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In *INTERSPEECH 2010, 11th annual conference of the international speech communication association, makuhari, chiba, japan, september 26-30, 2010* (pp. 1045–1048). Consulté sur http://www.isca-speech.org/archive/interspeech_2010/i10_1045.html

Mikolov, T., Kopecky, J., Burget, L., Glembek, O., & Cernocky, J. (2009). Neural network based language models for highly inflective languages. In *Proceedings of the 2009 ieee international conference on acoustics, speech and signal processing* (pp. 4725–4728). Washington, DC, USA : IEEE Computer Society. Consulté sur https://doi.org/10.1109/ICASSP.2009.4960686 doi: 10.1109/ICASSP.2009.4960686

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *CoRR*, *abs/1310.4546*. Consulté sur http://arxiv.org/abs/1310.4546

Mikolov, T., Yih, S. W.-t., & Zweig, G. (2013, May). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies (naacl-hlt-2013)* (Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2013) éd.). Association for Computational Linguistics. Consulté sur https://www.microsoft.com/en-us/research/publication/linguistic-regularities-in-continuous-space-word-representations/

Miller, G. A. (1995, novembre). Wordnet: A lexical database for english. *Commun. ACM*, *38*(11), 39–41. Consulté sur http://doi.acm.org/10.1145/219717.219748 doi: 10.1145/219717.219748

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, *2*, 559-572.

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *In emnlp*.

Redington, M., Chater, N., & Finch, S. (1998). Distributional information: A powerful cue for acquiring syntactic categories. *Cognitive Science*, *22*, 425-469.

Sahlgren, M. (2006). *The word-space model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces* (Thèse de doctorat non publiée). Stockholm University, Stockholm, Sweden.

Schwenk, H. (2007). Continuous space language models. *Comput. Speech Lang.*, *21*(3), 492–518. Consulté sur http://dx.doi.org/10.1016/j.csl.2006.09.003 doi: 10.1016/j.csl.2006.09.003

Turian, J., Ratinov, L., & Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics* (pp. 384–394). Stroudsburg, PA, USA : Association for Computational Linguistics. Consulté sur http://dl.acm.org/citation.cfm?id=1858681.1858721

van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*, 2579–2605. Consulté sur http://www.jmlr.org/papers/v9/vandermaaten08a.html